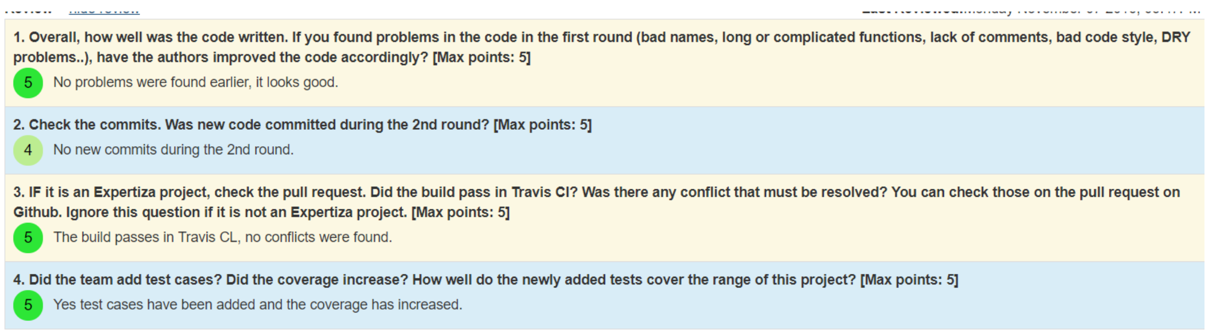
Independent Study – Report (CSC 630): Machine Learning for Peer Grading

Introduction:

Online courses or MOOCS have become very popular these days. Many students participate in these MOOCs and thus it becomes difficult for instructors to grade all the home works and assignments for all the students. Thus, they must resort to peer grading where students evaluate each other's work. The problem here is that not all students take peer grading very seriously. Some of them just do it because it is compulsory, yet others do it to get the bonus grades that are associated with it. This makes such reviews unreliable and the instructor must finally step in and decide the final grade. The goal of this project was to use Machine Learning to determine the reliability of peer reviews. Thus, the instructor would be able to separate the good reviews from the bad ones and this will help in reducing instructor involvement for the final grading.

Dataset:

The Data that we used for this project was obtained from Expertiza (an educational software being used at NC State University). Here peer reviews are used for 2 courses: CSC 517 and CSC 506. These peer reviews are graded by instructors and these scores are counted towards the final course score. Thus, we have a set of instructor graded peer reviews. These reviews consist of 2 major components: 1) Scores given for different criteria 2) Feedback justifying the score and advice on how improvements can be made. (Refer diagram bellow)



1. Overall, how well was the code written. If you found problems in the code in the first round (bad names, long or complicated functions, lack of comments, bad code style, DRY problems..), have the authors improved the code accordingly? [Max points: 5]

5 No problems were found earlier, it looks good.

2. Check the commits. Was new code committed during the 2nd round? [Max points: 5]

4 No new commits during the 2nd round.

3. IF it is an Expertiza project, check the pull request. Did the build pass in Travis CI? Was there any conflict that must be resolved? You can check those on the pull request on Github. Ignore this question if it is not an Expertiza project. [Max points: 5]

5 The build passes in Travis CL, no conflicts were found.

4. Did the team add test cases? Did the coverage increase? How well do the newly added tests cover the range of this project? [Max points: 5]

5 Yes test cases have been added and the coverage has increased.

We have instructor grades for all reviews for all assignments for the course CSC 517 from Spring of 2013 to Spring of 2017. For CSC 506, we have data from Spring 2015 to Fall of 2016. Each reviewer could do multiple reviews for every assignment, however the grades given were on a per assignment basis, thus we had 2 choices when grouping together data

Choice 1: Use the overall assignment score for a reviewer for each individual review done by him for that assignment.

Choice 2: Club together all reviews for a particular reviewer for a particular assignment,

Choice 1 clearly results in more data points around (11,600) while Choice 2 results in around 2500 data points. We tried using both datasets and realized that choice 1 performed better than choice 2 in general.

Goals:

The goal of our project was to use the above instructor graded data set to train a machine learning model to make predictions on the grades of new peer reviews. We used 2 main approaches here

1) **Regression**: We will try to predict the percentage score for each new review.
2) **Classification**: We will break the instructor scores into 5 classes: (scores < =50 (class 0), scores > 50 and <=70 (class 1), scores > 70 and <=85 (class 2), scores > 85 and <= 95 (class 3), scores > 95 (class 4))
   and use classification techniques to predict such classes on new reviews.

Steps undertaken:

Step 1 Gather data: The first step was putting together data, which was in raw form, into excel sheets so that it could be read into Python code and used for further analysis. Both the data sets (**Choice 1** and **Choice 2**) have been shared with Dr. Gehringer.

Step 2: Machine Learning Techniques:

1) Use of significant features: This technique extracts significant text based and score based features from the data and uses them for training the model. Some of the features we used were:

Text Based:

a) Number of words per sentence (wps)
b) Repetition score (rep_score): $\sum_{n=1}^{len(review)}(number\ of\ repeating\ ngrams) * n$
c) Count of words in a review belonging to different categories.
   I. Suggestion (sugc): should, must, might, could,….
   II. Location(locc): page, paragraph, sentence…
   III. Error(errc): error, mistake, typo, problem, …
   IV. Idead(idec): consider, mention
   V. Negatives (negc): fail, hard, bad, ….
   VI. Postives (posc): great, good, well….

For a full list refer paper by Wenting et al. [1]

d) Question-Answer correlation : For each word in the review question, try to find words that are similar to this word in the feedback. For finding similar words, we use wordnet.synset() [9] method in Python. This score is given as: $\frac{number\ of\ similar\ words\ in\ answer}{total\ number\ of\ words\ in\ question}$
e) Review Subjectivity (subjectivity): We used the subjectivity calculator from the textblob [2] software in Python.

Score Based:

a) Standard Deviation (std_dev): Compute standard deviation for all the scores given for a particular review.
b) Difference Score (diff_score): How different a reviewer's scores are from other scores (different reviewers) for the same submission. Here, we consider his score vector for a submission and compute the sum of its RMS differences from other score vectors for the same submission.

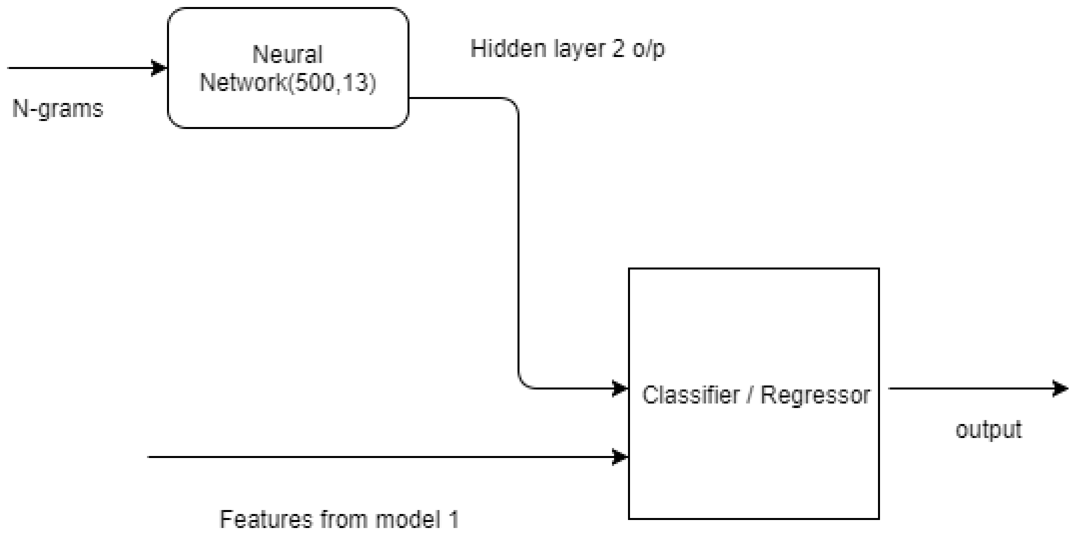
c) Similarity Score (simi_score): How similar a reviewer's scores for a particular submission are to his scores for other submissions in the same assignment. Here we add the pairwise RMS differences between different reviews (score vectors) done by the reviewer for a particular assignment.

(Features b and c are irrelevant when it comes to choice 2 of dataset and hence they are not used in that case.)

These features were fed to both classification and regression models. For regression, we used Decision trees and Neural Networks (hidden layer size = (5, 2)). For classification, we used Decision Trees, Naïve Bayes, and Neural Networks (hidden layer size = (5, 2)). We use 5- fold Cross Validation for both.

2) **N-Gram models:**
We make use of the traditional N-gram model, where we use frequencies of 1, 2 and 3 grams from text reviews as features. In order to reduce the number of features, we only consider N-grams with overall frequency greater than 100 in case of unigrams and 200 in case of bi and tri grams for choice 1 model. For choice 2 model, we consider the limit as 100 for all the 3. Again, we compute results using both Regression and Classification models as mentioned above.

3) **N-grams + Significant features:** Here, we try to combine both of the above methods for getting improved results. We initially train a neural network with our N-gram frequency features. This neural network has 2 hidden layers of sizes 500 and 13. Once this training is done, we use the output from the second hidden layer and combine it along with the significant features that we got as part of the 1st model to train a final classifier or a regression model. We use decision trees for both Classification and Regression models. A similar model was used by Ircing et al [3] for combining Textual and Speech features for the task of Native Language Identification.



4) **Text-CNN:** Convolutional Neural Networks have been known to work really well in image classification tasks. There have also been studies which have shown that CNN's work well in case of text classification tasks. Thus, we decided to try out these networks for our task. We

use the tutorial by Denny Britz [4] while making our model. Its structure is as follows. The code is a modification of the code from [5]

Structure:
Input data: (Max Length of Review) * 128 (chosen size of embedding)
Single Convolutional layer with filter sizes: 3, 4 and 5 (128 filters of each size)
1 max pooling layer (pooling size = Max Length of Review – filter_size -1), thus after max-pooling for each filter size, we have 128 values at the output (as we are choosing the maximum over each single filters' convolutions' output and there are 128 filters at each size.)
Thus final output of size: 128 * 3 = 384.
Dropout = 0.5
Connect to Softmax output of size 5 (number of classes)

For architecture diagram, please refer [4].

Results:

1) Significant Features:

Regression Analysis	Decision Tree	MLP (5, 2)
RMSE: (Choice 1)	18.23	37.80 (no convergence)
RMSE: (Choice 2)	20.77	26.09

Classification Analysis	Decision Tree	Naïve Bayes	MLP (5, 2)
Accuracy: (choice 1)	44.25	32.29	45.83
Accuracy: (choice 2)	33.79	28.96	40.48

2) N-grams:

Regression Analysis	Decision Tree	MLP (500)
RMSE: choice 1	17.311	17.59

RMSE: choice 2	19.264	23.799

Classification Analysis	Decision Tree	Naïve Bayes	MLP (500)
Accuracy: (choice 1)	42.6	36.84	46.26
Accuracy: (choice 2)	38.06	38.54	39.14

3) N-grams + Significant features:

Regression	Decision Tree
RMSE (choice 1)	16.375
RMSE (choice 2)	18.39

Classification	Decision Tree
Accuracy (choice 1)	48.33
Accuracy (choice 2)	38.33

4) Text Convolutional Neural Network:
Choice 1: Accuracy = 51.24%

Choice 2: Accuracy = 42.91%

Feature Selection: In order to find out which features contributed the most towards finding the final results, we performed feature selection using Univariate Feature Selection [6] and Recursive Feature Elimination with Cross Validation [7] techniques in the scikit learn package [8] in Python. Here is how the features were ranked:

a) Univariate Feature Selection:

Regression Tasks: simi_score, rep_score, wps, std_dev, subjectivity, pos_c, not_c

Classification Tasks: simi_score, rep_score, wps, subjectivity, pos_c, not_c, std_dev.

b) RFECV:
(Both Regression and Classifiaction): rep_score, simi_score, subjectivity, std_dev, diff_score, wps, pos_c, neg_c.

Thus, in general, score based features like simi_score and std_dev along with text based features like rep_score, subjectivity, wps, pos_c, and neg_c seem to be the best in terms of grade prediction.

(For all regression tasks Root Mean Squared Error was used as performance criteria, while for all classification tasks, accuracy was used.

From the results, it is clear that the dataset choice 2 yields worse results as compared to choice 1. Thus, for any further study, we will make use of the choice 1 dataset.)

Conclusion: As is apparent, the results obtained as part of this experiment were not very encouraging. The highest value we obtained in terms of accuracy was 51.24% for the Text CNN model, while in terms of RMSE, the best value obtained was 16.375 for the combined N-grams + Significant features model. Some of the areas where we could improve for further studies are:

1) Features:
    a. A better job could be done on pre-processing the given dataset. We could get rid of features that do not perform well according to our feature selection criteria
    b. Look for some other features that could contribute towards grade prediction
2) Text CNN:
    a. This was the method that gave the best results in terms of accuracy score. We could try to tinker with a few of the settings used here to obtain a better performance.
3) Rather than concentrating on 4 separate techniques, I could have worked on a single technique and tried to optimize it using techniques like Hyperparameter Optimization to get the best possible results.

GitHub Repository = https://github.com/SmokingSadhus/Machine-Learning-for-Peer-Grading

References

[1] W. Xiong and D. Litman, "Understanding Differences in Perceived Peer-Review Helpfulness using Natural Language Processing," in *Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, Portland, 2011.

[2] S. Loria, "textblob," in *http://textblob.readthedocs.io/en/dev/*.

[3] P. Ircing, J. Svec, Z. Zajic, B. Hladka and M. Holub, "Combining Textual and Speech Features in the NLI Task Using State-of-the-Art Machine Learning Techniques," Copenhagen, 2017.

[4] D. Britz, "Implementing a CNN for Text Classification in TensorFlow," in *http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/*.

[5] D. Britz, "cnn-text-classification-tf," in *https://github.com/dennybritz/cnn-text-classification-tf*.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Univariate Feature Selection," *Journal of Machine Learning Research: http://scikit-*

learn.org/stable/auto_examples/feature_selection/plot_feature_selection.html#sphx-glr-auto-examples-feature-selection-plot-feature-selection-py, vol. 12, pp. 2825-2830, 2011.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Recursive Feature Elimination with Cross Validation," *Journal of Machine Learning Research: http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py,* vol. 12, pp. 2825-2830, 2011.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research,* vol. 12, pp. 2825--2830, 2011.

[9] Bird, Steven, Edward Loper, Ewan Klein (2009), Natural Language Processing with Python, O'Reilly Media http://www.nltk.org/howto/wordnet.html